



INFERENCIA Y MODELOS ESTADÍSTICOS

Jacqueline Köhler C. y José Luis Jara V.



CAPÍTULO 12. REMUESTREO

Los **métodos basados en remuestreo** son una buena alternativa a emplear cuando necesitamos inferir sobre parámetros distintos a la media o la proporción, o bien cuando no se cumplen las condiciones requeridas por las pruebas ya conocidas. Además, algunos de estos métodos son más precisos que los tradicionales. Pese a estas ventajas, los métodos basados en remuestreo realizan enormes cantidades de cálculos, por lo que en la práctica requieren de herramientas de software para su aplicación. Si bien existen métodos de remuestreo paramétricos y semiparamétricos, en este capítulo abordaremos las principales técnicas de remuestreo no paramétricas, basándonos en las ideas descritas por Amat Rodrigo (2016) y Hesterberg y col. (2003).

12.1 BOOTSTRAPPING

A partir de lo que hemos aprendido hasta ahora, ya tenemos bastante claro que, en estadística, el ideal es contar con varias muestras grandes. Pero muchas veces solo disponemos de una muestra bastante pequeña. Sin embargo, si esta muestra es representativa de la población, esperaríamos que las observaciones que ella contiene aparecieran con frecuencias similares a las de la población. El método de **bootstrapping** se construye en torno a esta idea y, en términos generales, sigue los siguientes pasos:

1. Crear una gran cantidad B de nuevas muestras (cientos o miles) a partir de la muestra original. Cada muestra debe tener el mismo tamaño que la original y se construye mediante **muestreo con reposición**. Esto quiere decir que, al seleccionar un elemento de la muestra original, se devuelve a ella antes de tomar el siguiente, por lo que podría ser reelegido.
2. Calcular la distribución bootstrap y obtener el estadístico de interés para cada una de las muestras.
3. Usar la distribución bootstrap, la cual entrega información acerca de la forma, el centro y la variabilidad de la distribución muestral del estadístico de interés.

A los lectores atentos les habrá llamado la atención que, a diferencia de las pruebas y procedimientos anteriores, el segundo paso del método de bootstrapping habla de un **estadístico de interés** en lugar de la media o la proporción (como las pruebas estudiadas hasta ahora). Esto se debe a que, en general, puede aplicarse para casi **cualquier estadístico**.

Esta técnica, además de contrastar hipótesis, permite construir intervalos de confianza para el parámetro estimado de la población.

12.1.1 Bootstrapping para una muestra

Supongamos que la investigadora Helen Chufe desea evaluar un nuevo algoritmo de clasificación y determinar el tiempo promedio de ejecución (en milisegundos) para instancias de tamaño fijo del problema. Para ello ha realizado pruebas con 10 instancias del problema y registrado los tiempos de ejecución, presentados en la tabla 12.1. La figura 12.1 muestra la distribución del tiempo de ejecución para la muestra.

Evidentemente, la muestra es pequeña ($n = 10$) y su distribución está fuertemente desviada hacia la izquierda, por lo que Chufe ha decidido emplear bootstrapping como alternativa para enfrentar estos datos problemáticos.

Instancia	1	2	3	4	5	6	7	8	9	10
Tiempo (ms)	79	75	84	75	94	82	76	90	79	88

Tabla 12.1: tiempo de ejecución para cada instancia de la muestra.

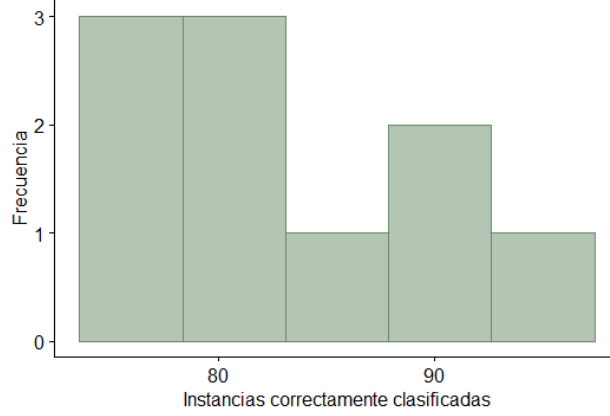


Figura 12.1: distribución del tiempo de ejecución para la muestra.

Para ilustrar el proceso paso a paso, consideremos inicialmente $B = 10$ remuestreos y calculemos la media para cada uno. La tabla 12.2 muestra en cada columna una de las muestras obtenidas, con sus respectivas medias en la última fila.

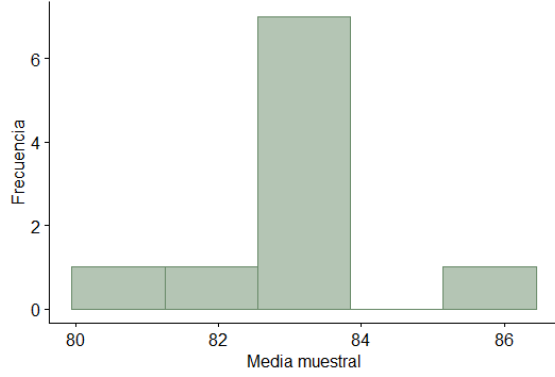
Original	Bt 1	Bt 2	Bt 3	Bt 4	Bt 5	Bt 6	Bt 7	Bt 8	Bt 9	Bt 10
79	84	84	84	94	94	94	76	82	75	79
75	94	75	82	88	75	75	88	88	82	79
84	79	82	84	90	90	94	79	79	94	94
75	79	88	79	90	82	94	76	75	94	82
94	88	79	79	90	82	76	84	75	79	84
82	75	84	79	75	76	75	75	79	76	82
76	88	82	84	75	76	79	75	90	88	94
90	88	79	79	75	82	75	79	88	88	79
79	84	79	90	94	90	88	75	75	79	75
88	75	94	88	82	88	76	94	90	82	82
82,2	83,4	82,6	82,8	85,3	83,5	82,6	80,1	82,1	83,7	83,0

Tabla 12.2: muestra original y remuestreos de bootstrap

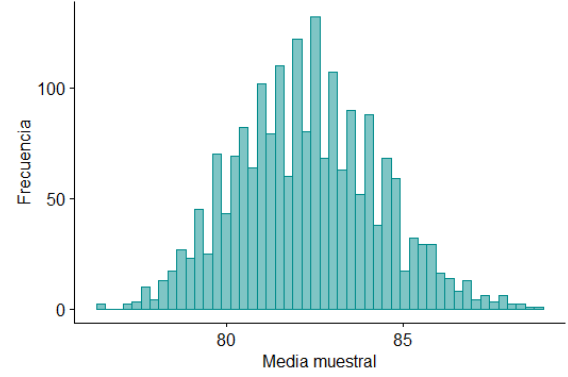
La figura 12.2 muestra la distribución bootstrap de la media para los 10 remuestreos del ejemplo (figura 12.2a) y para 2.000 remuestreos (figura 12.2b). En ella podemos ver claramente que, a medida que la cantidad de muestras bootstrap crece, la distribución bootstrap de la media se asemeja cada vez más a la distribución normal, por lo que se acerca a la forma que esperaríamos para la distribución muestral. La figura 12.2b también nos da una idea acerca de la variabilidad de las medias de los diferentes remuestreos. Sin embargo, podemos conocer mejor esta variabilidad calculando el error estándar, dado por la ecuación 12.1, donde \bar{x}_i^* es la media del i -ésimo remuestreo y B es la cantidad de remuestreos realizados.

$$SE_{b,\bar{x}} = \sqrt{\frac{1}{B-1} \cdot \sum_{i=1}^B \left(\bar{x}_i^* - \frac{1}{B} \cdot \sum_{i=1}^B \bar{x}_i^* \right)^2} \quad (12.1)$$

Fijémonos en que la subexpresión $\frac{1}{B} \cdot \sum_{i=1}^B \bar{x}_i^*$ de la ecuación 12.1 corresponde al promedio de las medias de



(a) 10 remuestreos.



(b) 2.000 remuestreos.

Figura 12.2: distribución bootstrap de la media

los remuestreos (media de la distribución bootstrap), por lo que la subexpresión $\sum_{i=1}^B \left(\bar{x}_i^* - \frac{1}{B} \cdot \sum_{i=1}^B \bar{x}_i^* \right)^2$ es, a su vez, la suma de las desviaciones cuadradas, con lo que resulta evidente la semejanza con el cálculo del error estándar para la media de una muestra presentado en el capítulo 4:

$$SE_{\bar{x}} = \frac{s}{\sqrt{n}}$$

Para el ejemplo con $B = 10$, entonces, tenemos que la media de la distribución bootstrap es:

$$\frac{1}{B} \cdot \sum_{i=1}^B \bar{x}_i^* = 83,4 + 82,6 + 82,8 + 85,3 + 83,5 + 82,6 + 80,1 + 82,1 + 83,7 + 83,0 = 82,91$$

Con lo que la suma de las desviaciones cuadradas es:

$$\begin{aligned} \sum_{i=1}^B (\bar{x}_i^* - 82,91)^2 &= (83,4 - 82,91)^2 + (82,6 - 82,91)^2 + (82,8 - 82,91)^2 + \\ &+ (85,3 - 82,91)^2 + (83,5 - 82,91)^2 + (82,6 - 82,91)^2 + (80,1 - 82,91)^2 + \\ &+ (82,1 - 82,91)^2 + (83,7 - 82,91)^2 + (83,0 - 82,91)^2 = 15,689 \end{aligned}$$

En consecuencia, el error estándar de la distribución bootstrap es:

$$SE_{B,\bar{x}} = \sqrt{\frac{1}{B-1} \cdot 15,689} = 1,320$$

Otra medida que suele emplearse para la distribución bootstrap es el **sesgo**, que indica cuánto se aleja el estadístico de interés de la muestra original (θ) de la media de la distribución bootstrap, como muestra la ecuación 12.2.

$$sesgo = \theta - \frac{1}{B} \cdot \sum_{i=1}^B \bar{x}_i^* \quad (12.2)$$

Para el ejemplo:

$$sesgo = \bar{x} - \frac{1}{B} \cdot \sum_{i=1}^B \bar{x}_i^* = 82,2 - 82,91 = -0,71$$

Ahora que ya conocemos la distribución bootstrap para la media, podemos entonces **construir un intervalo de confianza para la media de la población**, para lo que abordaremos diferentes alternativas.

Cuando la distribución bootstrap se asemeja a la normal y el sesgo es pequeño en comparación con el estimador calculado (como en este caso), podemos construir el intervalo de confianza usando la distribución t , del mismo modo que conocimos en el capítulo 4, como muestra la ecuación 12.3, usando el error estándar de la distribución bootstrap. El valor crítico de t , t^* , se obtiene para el nivel de significación establecido para el estudio y $\nu = n - 1$ grados de libertad (no olvidemos que n es el tamaño de la muestra original).

$$\theta \pm t^* \cdot SE_{B,\bar{x}} \quad (12.3)$$

Así, si consideramos para este ejemplo un nivel de significación $\alpha = 0,01$, el valor crítico de t (para dos colas) con 9 grados de libertad es $t^* = 3,25$. En consecuencia, el intervalo de confianza resultante para la media de la población es:

$$82,2 \pm 3,25 \cdot 1,320 = (77,910; 86,490)$$

Otra alternativa cuando la distribución bootstrap se asemeja a la normal, que tiene en cuenta posibles asimetrías, es construir el intervalo de confianza en base a cuantiles. En este caso, para $\alpha = 0,01$, los límites del intervalo están dados por los percentiles 1 y 99 de la distribución bootstrap:

$$(80,280; 85,156)$$

Cuando los intervalos de confianza obtenidos por ambos métodos son muy diferentes, es clara señal de que no podemos asumir que la distribución bootstrap se asemeja a la normal. En general, lo más recomendable es usar otro esquema, llamado BCa (del inglés *bias-corrected accelerated*), es decir, con sesgo corregido y acelerado. No se detalla aquí el procedimiento, pues requiere el empleo de software.

Desde luego, es inviable usar bootstrapping sin software. R ofrece el paquete `boot`, con las funciones `boot(data, statistic, R)` para generar la distribución bootstrap y `boot.ci(boot.out, conf, type)` para calcular los intervalos de confianza, donde:

- **data**: el conjunto de datos. En caso de matrices y data frames, se considera cada fila como una observación con múltiples variables.
- **statistic**: función que se aplica a los datos y devuelve un vector con el (o los) estadístico(s) de interés.
- **R**: cantidad de remuestreos bootstrap (B).
- **boot.out**: objeto de la clase `boot`, generado por la función `boot()`.
- **conf**: nivel de confianza ($1 - \alpha$).
- **type**: string o vector que indica los tipos de intervalo de confianza a construir (“**norm**” para el basado en la distribución normal, “**perc**” para el basado en los percentiles y “**bca**” para el método recomendado).

Debemos mencionar que la función `boot()` puede recibir otros muchos argumentos, los cuales escapan al alcance de los contenidos aquí expuestos. El script 12.1 construye intervalos de confianza mediante bootstrapping para el ejemplo, con $B = 2000$ y manteniendo el nivel de significación $\alpha = 0,01$. En las líneas 15–17 se construye la función para el estadístico de interés (en este caso la media), que luego usa la función `boot()` para generar la distribución bootstrap (líneas 19–20), obteniéndose el resultado que se presenta en la figura 12.3.

Podemos ver gráficamente esta distribución mediante un histograma y un gráfico Q-Q (figura 12.4), gracias a la llamada a la función `plot()` con el resultado entregado por `boot()` como argumento (línea 23).

ORDINARY NONPARAMETRIC BOOTSTRAP

```
Call:
boot(data = muestra, statistic = media, R = B)
```

```
Bootstrap Statistics :
      original  bias      std. error
t1*      82.2 0.06125      1.98329
```

Figura 12.3: distribución bootstrap generada mediante `boot()` para la media.

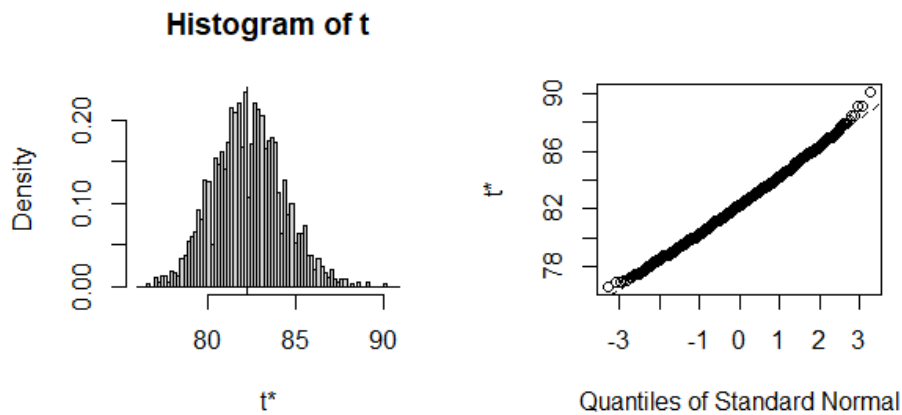


Figura 12.4: histograma y gráfico Q-Q de la distribución bootstrap generada mediante `boot()` para la media.

En las líneas 26–39 se muestra el uso de `boot.ci()` para construir los intervalos de confianza mediante diferentes métodos, obteniéndose los siguientes resultados:

- Intervalo de confianza usando aproximación normal: (77, 03; 87, 25).
- Intervalo de confianza usando percentiles: (77, 4; 87, 7).
- Intervalo de confianza BCa: (77, 48; 87, 90).

Las líneas 42–45 muestran otra alternativa para construir la distribución bootstrap por medio del paquete `bootES`, que ofrece la función `bootES(data, R, ci.type, ci.conf, plot, ...)`. Esta función realiza internamente una llamada a la función `boot()` descrita en los párrafos precedentes, pero no requiere implementar previamente la función para el cálculo de la media. Debemos tener en cuenta que aquí solo se muestran algunos de los argumentos, a saber:

- **data**: conjunto de datos.
- **R**: cantidad de remuestreos bootstrap (B).
- **ci.type**: tipo de intervalo de confianza a construir (opcional), con las mismas opciones descritas para `boot.ci()`.
- **ci.conf**: nivel de significación para el intervalo de confianza (opcional, por defecto 0.95).
- **plot**: por defecto con valor `FALSE`, cuando es `TRUE` genera una figura con el histograma y el gráfico Q-Q de la distribución bootstrap.
- **...**: permite pasar otros argumentos para la función `boot()` subyacente.

Las figuras 12.5 y 12.6 muestran los resultados obtenidos, ligeramente diferentes a los anteriores. A partir de

estos últimos podemos concluir que tenemos 95 % de confianza de que el algoritmo tarda entre 77,48 ms y 87,90 ms en ejecutar las instancias del tamaño seleccionado.

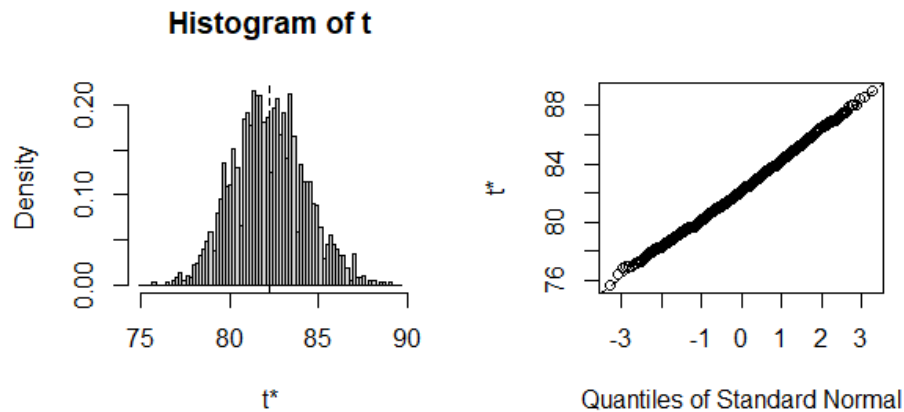


Figura 12.5: histograma y gráfico Q-Q de la distribución bootstrap generada mediante `bootES()` para la media.

99.00% bca Confidence Interval, 2000 replicates				
Stat	CI (Low)	CI (High)	bias	SE
82.200	77.482	87.900	0.061	1.983

Figura 12.6: distribución bootstrap e intervalo de confianza para la media de la población generada mediante `bootES()`.

Script 12.1: construcción de un intervalo de confianza para la media poblacional mediante bootstrapping.

```

1 library(boot)
2 library(bootES)
3
4 # Crear muestra inicial, mostrar su histograma y calcular la media.
5 muestra <- c(79, 75, 84, 75, 94, 82, 76, 90, 79, 88)
6 datos <- data.frame(muestra)
7
8 # Establecer cantidad de remuestreos y nivel de significación.
9 B = 2000
10 alfa <- 0.01
11
12 cat("Paquete boot\n")
13
14 # Construir distribución bootstrap usando el paquete boot.
15 media <- function(valores, i) {
16   mean(valores[i])
17 }
18
19 set.seed(432)
20 distribucion_b <- boot(muestra, statistic = media, R = B)
21 print(distribucion_b)
22
23 # Graficar distribución bootstrap.
24 print(plot(distribucion_b))
25

```



```

26 # Construir intervalos de confianza.
27 intervalo_t <- boot.ci(distribucion_b, conf = 1 - alfa, type = "norm")
28
29 cat("\n\nIntervalo de confianza usando distribución t:\n")
30 print(intervalo_t)
31
32 intervalo_per <- boot.ci(distribucion_b, conf = 1 - alfa, type = "perc")
33
34 cat("\n\nIntervalo de confianza usando percentiles:\n")
35 print(intervalo_per)
36
37 intervalo_bca <- boot.ci(distribucion_b, conf = 1 - alfa, type = "bca")
38
39 cat("\n\nIntervalo de confianza BCa:\n")
40 print(intervalo_bca)
41
42 # Construir distribución bootstrap usando el paquete bootES.
43 set.seed(432)
44
45 distribucion_bootstrapES <- bootES(muestra, R = B, ci.type = "bca",
46                                   ci.conf = 1 - alfa, plot = TRUE)
47
48 print(distribucion_bootstrapES)

```

Supongamos ahora que Helen desea hacer una prueba de hipótesis para ver si el tiempo promedio de ejecución del algoritmo para instancias del tamaño seleccionado es mayor a 75 milisegundos. Así, tenemos que:

Denotando como μ al tiempo medio que tarda el algoritmo de Helen para resolver instancias de tamaño fijo del problema, entonces:

$H_0: \mu = 75$ [ms]

$H_A: \mu > 75$ [ms]

El contraste de hipótesis requiere siempre generar la distribución centrada en el valor nulo para, a partir de ella, obtener el valor p. Sabemos que la distribución bootstrap se centra *alrededor* del valor observado, por lo que debemos desplazarla para cumplir con esta condición. Para lograrlo, simplemente necesitamos restar a cada observación de la distribución bootstrap la diferencia entre su valor promedio y el valor nulo.

Para calcular el valor p, seguimos la fórmula señalada en la ecuación 12.4, donde:

- r : cantidad de observaciones en la distribución bootstrap (desplazada) a lo menos tan extremas como el estadístico observado.
- B : cantidad de repeticiones bootstrap consideradas en la simulación.

$$p = \frac{r + 1}{B + 1} \quad (12.4)$$

Tras hacer la prueba (script 12.2), obtenemos que $p = 0,001$, menor que el nivel de significación, por lo que la evidencia es suficientemente fuerte para rechazar la hipótesis nula en favor de la hipótesis alternativa. En consecuencia, concluimos con 99,5% de confianza que el tiempo de ejecución promedio del algoritmo para instancias del tamaño seleccionado supera los 75 milisegundos.

Script 12.2: inferencia sobre la media de una muestra con bootstrapping.

```
1 library(boot)
2
3 set.seed(432)
4
5 # Crear muestra inicial, mostrar su histograma y calcular la media.
6 muestra <- c(79, 75, 84, 75, 94, 82, 76, 90, 79, 88)
7 valor_observado <- mean(muestra)
8 datos <- data.frame(muestra)
9
10 # Construir distribución bootstrap.
11 B <- 2000
12
13 media <- function(valores, i) {
14   mean(valores[i])
15 }
16
17 distribucion_b <- boot(muestra, statistic = media, R = B)
18
19 # Desplazar la distribución bootstrap para que se centre en
20 # el valor nulo.
21 valor_nulo <- 75
22 desplazamiento <- mean(distribucion_b[["t"]]) - valor_nulo
23 distribucion_nula <- distribucion_b[["t"]] - desplazamiento
24
25 # Determinar el valor p.
26 p <- (sum(distribucion_nula > valor_observado) + 1) / (B + 1)
27 cat("Valor p:", p)
```

12.1.2 Bootstrapping para dos muestras independientes

El proceso para comparar dos poblaciones mediante bootstrapping es similar al que ya conocimos para una única población. Si tenemos dos muestras independientes A y B provenientes de dos poblaciones diferentes, de tamaños n_A y n_B respectivamente, los pasos a seguir son:

1. Fijar la cantidad B de repeticiones bootstrap.
2. En cada repetición, hacer un remuestreo con reposición de tamaño n_A a partir de la muestra A y otro de tamaño n_B a partir de la muestra B .
3. En cada repetición, calcular el estadístico de interés para generar la distribución bootstrap.
4. Construir el intervalo de confianza para el estadístico de interés.

Supongamos que una Universidad desea estudiar la diferencia entre las calificaciones finales de hombres y mujeres que rinden una asignatura inicial de programación por primera vez. Para ello, disponen de las notas (en escala de 1,0 a 7,0) de 27 hombres y 19 mujeres:

- Hombres: 1,3; 1,5; 1,6; 1,7; 1,7; 1,9; 2,3; 2,4; 2,6; 2,6; 2,7; 2,8; 3,2; 3,7; 4,1; 4,4; 4,5; 4,8; 5,2; 5,2; 5,3; 5,5; 5,5; 5,6; 5,6; 5,7; 5,7
- Mujeres: 3,5; 3,6; 3,8; 4,3; 4,5; 4,5; 4,9; 5,1; 5,3; 5,3; 5,5; 5,8; 6,0; 6,3; 6,3; 6,4; 6,4; 6,6; 6,7

Tras aplicar pruebas de Shapiro-Wilk (figura 12.7), los investigadores han comprobado que las notas de los varones no siguen una distribución normal, por lo que han decidido usar bootstrapping para la prueba de hipótesis, con un nivel de significación $\alpha = 0,05$ y $B = 9999$ repeticiones.

```
Shapiro-Wilk normality test

data:  hombres
W = 0.88357, p-value = 0.005742
```

```
Shapiro-Wilk normality test

data:  mujeres
W = 0.93022, p-value = 0.1748
```

Figura 12.7: pruebas de normalidad de Shappiro-Wilk para ambas muestras.

La media observada (en la muestra original) para la calificación final de las mujeres es $\bar{x}_m = 5,305$, mientras que para los hombres es $\bar{x}_h = 3,670$. Así, la diferencia observada es $\bar{x}_h - \bar{x}_m = -1,635$.

La distribución bootstrap de la diferencia de medias se asemeja a la normal (figura 12.8), con media $\bar{x} = -1,628$ y desviación estándar $s = 0,377$.

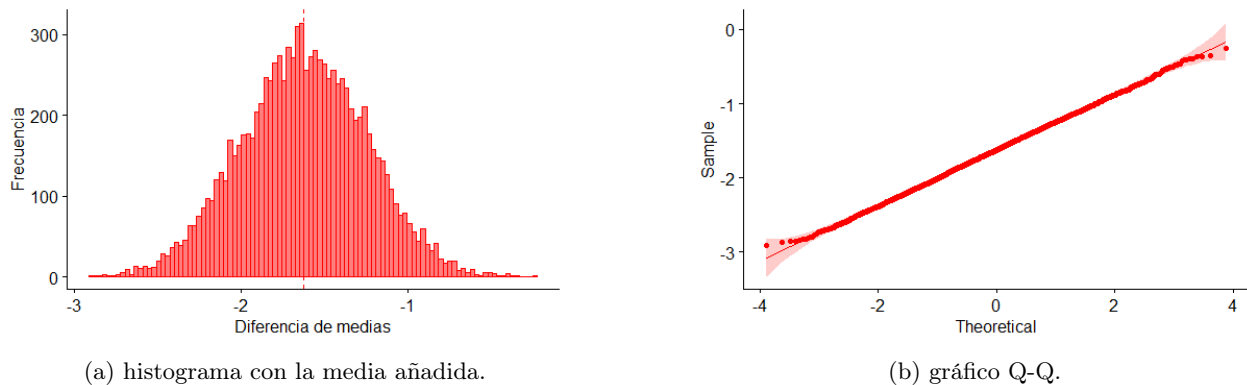


Figura 12.8: distribución bootstrap de la diferencia de medias.

Al construir el intervalo de confianza mediante el método BCa para la distribución bootstrap, R nos entrega como resultado el intervalo $(-2,372; -0,894)$. En consecuencia, concluimos con 95 % de confianza que las mujeres tienen, en promedio, mejor calificación final que los hombres, con una diferencia de entre 0,894 y 2,372 puntos.

El script 12.3 muestra el desarrollo de este ejemplo en R, el cual usa la función `two.boot(sample1, sample2, FUN, R)` del paquete `simpleboot`, donde:

- `sample1`, `sample2`: muestras originales.
- `FUN`: función que, para cada muestra, calcula el estadístico de interés θ .
- `R`: cantidad de remuestreos con repetición.

Esta función opera generando remuestreos para cada una de las muestras originales, y calculando en cada iteración el estadístico $\theta_1 - \theta_2$, donde los subíndices señalan la muestra correspondiente.

Script 12.3: bootstrapping para la diferencia de medias.

```
1 library(simpleboot)
2 library(boot)
3 library(ggpubr)
4
5 set.seed(432)
6
```

```

7 # Ingresar datos originales
8 hombres <- c(1.3, 1.5, 1.6, 1.7, 1.7, 1.9, 2.3, 2.4, 2.6, 2.6, 2.7,
9             2.8, 3.2, 3.7, 4.1, 4.4, 4.5, 4.8, 5.2, 5.2, 5.3, 5.5,
10            5.5, 5.6, 5.6, 5.7, 5.7)
11
12 mujeres <- c(3.5, 3.6, 3.8, 4.3, 4.5, 4.5, 4.9, 5.1, 5.3, 5.3, 5.5,
13            5.8, 6.0, 6.3, 6.3, 6.4, 6.4, 6.6, 6.7)
14
15 n_hombres <- length(hombres)
16 n_mujeres <- length(mujeres)
17
18 sexo <- c(rep("Hombre", n_hombres), rep("Mujer", n_mujeres))
19 nota <- c(hombres, mujeres)
20 datos <- data.frame(nota, sexo)
21
22 # Comprobar normalidad de las muestras.
23 print(shapiro.test(hombres))
24 print(shapiro.test(mujeres))
25
26 # Calcular la diferencia observada entre las medias muestrales.
27 media_hombres <- mean(hombres)
28 media_mujeres <- mean(mujeres)
29 diferencia_observada <- media_hombres - media_mujeres
30
31 cat("diferencia observada:", media_hombres - media_mujeres, "\n\n")
32
33 # Establecer el nivel de significación.
34 alfa <- 0.05
35
36 # Crear la distribución bootstrap.
37 B <- 9999
38 distribucion_bootstrap <- two.boot(hombres, mujeres, FUN = mean, R = B)
39
40 # Examinar la distribución bootstrap.
41 valores <- data.frame(distribucion_bootstrap$t)
42 colnames(valores) <- "valores"
43
44 histograma <- gghistogram(valores, x = "valores", color = "red",
45                          fill = "red", bins = 100,
46                          xlab = "Diferencia de medias",
47                          ylab = "Frecuencia", add = "mean")
48
49 print(histograma)
50
51 qq <- ggqqplot(valores, x = "valores", color = "red")
52 print(qq)
53
54 cat("Distribución bootstrap:\n")
55 cat("\tMedia:", mean(valores$valores), "\n")
56 cat("\tDesviación estándar:", sd(valores$valores), "\n\n")
57
58 # Construir el intervalo de confianza.
59 intervalo_bca <- boot.ci(distribucion_bootstrap, conf = 1 - alfa,
60                        type = "bca")
61
62 print(intervalo_bca)

```

Supongamos ahora que el estudio del ejemplo desea determinar, con un nivel de significación $\alpha = 0,05$, si la diferencia entre las calificaciones finales de hombres y mujeres es igual a 1,5 puntos. Para ello, formulamos

las siguientes hipótesis:

Sean μ_h y μ_m las calificaciones finales de hombres y mujeres, respectivamente, que rinden una asignatura inicial de programación por primera vez en la Universidad en estudio, entonces:

$H_0: \mu_h - \mu_m = 1,5$

$H_A: \mu_h - \mu_m \neq 1,5$

Tras aplicar bootstrapping para la prueba de hipótesis (script 12.4), obtenemos un valor p de $p = 0,364$, superior al nivel de significación, por lo que fallamos al rechazar la hipótesis nula. En consecuencia, concluimos con 95 % de confianza que la diferencia en la calificación final entre hombres y mujeres es de 1,5 puntos.

Script 12.4: bootstrapping para inferir acerca de la diferencia de medias.

```
1 library(simpleboot)
2 library(boot)
3 library(ggpubr)
4
5 set.seed(432)
6
7 # Ingresar datos originales
8 hombres <- c(1.3, 1.5, 1.6, 1.7, 1.7, 1.9, 2.3, 2.4, 2.6, 2.6, 2.7,
9             2.8, 3.2, 3.7, 4.1, 4.4, 4.5, 4.8, 5.2, 5.2, 5.3, 5.5,
10            5.5, 5.6, 5.6, 5.7, 5.7)
11
12 mujeres <- c(3.5, 3.6, 3.8, 4.3, 4.5, 4.5, 4.9, 5.1, 5.3, 5.3, 5.5,
13            5.8, 6.0, 6.3, 6.3, 6.4, 6.4, 6.6, 6.7)
14
15 n_hombres <- length(hombres)
16 n_mujeres <- length(mujeres)
17
18 sexo <- c(rep("Hombre", n_hombres), rep("Mujer", n_mujeres))
19 nota <- c(hombres, mujeres)
20 datos <- data.frame(nota, sexo)
21
22 # Calcular la diferencia observada entre las medias muestrales.
23 media_hombres <- mean(hombres)
24 media_mujeres <- mean(mujeres)
25 valor_observado <- media_hombres - media_mujeres
26
27 # Crear la distribución bootstrap.
28 B <- 9999
29 valor_nulo <- 1.5
30 distribucion_bootstrap <- two.boot(hombres, mujeres, FUN = mean, R = B)
31 desplazamiento <- mean(distribucion_bootstrap[["t"]]) - valor_nulo
32 distribucion_nula <- distribucion_bootstrap[["t"]] - desplazamiento
33
34 # Determinar el valor p.
35 p <- (sum(abs(distribucion_nula) > abs(valor_observado)) + 1) / (B + 1)
36 cat("Valor p:", p)
```

12.1.3 Bootstrapping para dos muestras pareadas

En este caso, el procedimiento resulta muy sencillo. A partir de las dos muestras originales, se crea una nueva muestra con la diferencia entre ambas, y luego se realiza el proceso especificado para la construcción de un intervalo de confianza que ya conocimos para el caso de una única muestra.

Supongamos ahora, que la Universidad del ejemplo anterior desea saber si existe diferencia entre las calificaciones obtenidas en la primera y la segunda prueba de un curso inicial de programación. Para ello, dispone de las calificaciones (en escala de 1,0 a 7,0) obtenidas en ambas pruebas para una muestra de 20 estudiantes, como muestra la tabla 12.3. Han decidido llevar a cabo el estudio mediante bootstrapping con $B = 3999$ repeticiones y un nivel de significación $\alpha = 0,05$, para lo cual han creado en R el script 12.5, obteniendo los resultados que se presentan en la figura 12.9.

Alumno	Prueba 1	Prueba 2
1	3,5	5,2
2	2,7	5,1
3	1,0	5,9
4	1,8	4,8
5	1,6	1,4
6	4,3	2,3
7	5,8	6,8
8	6,4	5,3
9	3,9	3,1
10	4,3	3,8
11	3,4	4,6
12	5,3	1,2
13	5,8	3,9
14	5,3	2,0
15	2,0	1,7
16	1,3	3,3
17	4,0	6,0
18	5,3	4,8
19	1,6	6,9
20	3,6	1,3

Tabla 12.3: calificaciones de los estudiantes en la primera y la segunda prueba de un curso inicial de programación.

```

95.00% bca Confidence Interval, 3999 replicates
Stat      CI (Low)      CI (High)      bias      SE
0.325      -0.656      1.439      0.001      0.541

```

Figura 12.9: intervalo de confianza BCa para la media de las diferencias.

A partir del resultado anterior, concluimos con 95 % de confianza que la diferencia de las medias para las calificaciones de la primera y la segunda evaluación se encuentra en el intervalo $(-0,656; 1,439)$, por lo que no hay una diferencia estadísticamente significativa.

Script 12.5: bootstrapping para la media de las diferencias.

```

1 library(bootES)
2
3 set.seed(432)
4
5 # Ingresar datos originales.
6 alumno <- 1:20
7
8 prueba_1 <- c(3.5, 2.7, 1.0, 1.8, 1.6, 4.3, 5.8, 6.4, 3.9, 4.3, 3.4,
9              5.3, 5.8, 5.3, 2.0, 1.3, 4.0, 5.3, 1.6, 3.6)
10
11 prueba_2 <- c(5.2, 5.1, 5.9, 4.8, 1.4, 2.3, 6.8, 5.3, 3.1, 3.8, 4.6,
12              1.2, 3.9, 2.0, 1.7, 3.3, 6.0, 4.8, 6.9, 1.3)

```

```

13
14 # Establecer nivel de significación.
15 alfa <- 0.05
16
17 # Calcular la diferencia entre ambas observaciones.
18 diferencia <- prueba_2 - prueba_1
19
20 # Generar la distribución bootstrap y su intervalo de confianza.
21 B <- 3999
22
23 distribucion_bootstrapES <- bootES(diferencia, R = B, ci.type = "bca",
24                                   ci.conf = 1 - alfa, plot = FALSE)
25
26 print(distribucion_bootstrapES)

```

Ahora la Universidad del ejemplo desea saber si la diferencia entre las calificaciones obtenidas en la primera y la segunda prueba de un curso inicial de programación es de 5 décimas. Así, considerando un nivel de significación $\alpha = 0,05$, los investigadores formulan las siguientes hipótesis:

$$H_0: \mu_{dif} = 0,5$$

$$H_0: \mu_{dif} \neq 0,5$$

Tras efectuar la prueba de hipótesis mediante bootstrapping (script 12.6) obtienen un valor p de $p = 0,573$, por lo que la evidencia no es suficientemente fuerte como para rechazar la hipótesis nula. En consecuencia, los investigadores concluyen con 95 % de confianza que la diferencia de las calificaciones obtenidas en ambas evaluaciones es de 5 décimas.

Script 12.6: bootstrapping para inferir acerca de la media de las diferencias.

```

1 library(bootES)
2
3 set.seed(432)
4
5 # Ingresar datos originales.
6 alumno <- 1:20
7
8 prueba_1 <- c(3.5, 2.7, 1.0, 1.8, 1.6, 4.3, 5.8, 6.4, 3.9, 4.3, 3.4,
9              5.3, 5.8, 5.3, 2.0, 1.3, 4.0, 5.3, 1.6, 3.6)
10
11 prueba_2 <- c(5.2, 5.1, 5.9, 4.8, 1.4, 2.3, 6.8, 5.3, 3.1, 3.8, 4.6,
12              1.2, 3.9, 2.0, 1.7, 3.3, 6.0, 4.8, 6.9, 1.3)
13
14 # Establecer nivel de significación.
15 alfa <- 0.05
16
17 # Calcular la diferencia entre ambas observaciones.
18 diferencia <- prueba_2 - prueba_1
19
20 # Calcular la media observada de las diferencias.
21 valor_observado <- mean(diferencia)
22
23 # Generar la distribución bootstrap y su intervalo de confianza.
24 B <- 3999
25 valor_nulo <- 0.5
26
27 distribucion_bootstrapES <- bootES(diferencia, R = B, ci.type = "bca",
28                                   ci.conf = 1 - alfa, plot = FALSE)
29
30 distribucion_nula <- distribucion_bootstrapES[["t"]] - valor_nulo
31

```

```

32
33 # Determinar el valor p.
34 p <- (sum(abs(distribucion_nula) > abs(valor_observado)) + 1) / (B + 1)
35 cat("Valor p:", p)

```

12.2 PRUEBAS DE PERMUTACIONES

En el capítulo 8 conocimos la prueba exacta de Fisher, la cual obtiene un valor p exacto tras calcular todas las permutaciones de los datos con iguales valores marginales en una tabla de contingencia como alternativa para muestras pequeñas de la prueba y considerar únicamente aquellas permutaciones que ocurren con igual o menor probabilidad que la obtenida para los datos del estudio.

La prueba exacta de Fisher es lo que se conoce como una **prueba exacta de permutaciones**, cuyo único requisito es la **intercambiabilidad**: si se cumple la hipótesis nula, todas las permutaciones pueden ocurrir con igual probabilidad. En la práctica, este tipo de métodos puede emplearse para diversos estadísticos, tales como la proporción, la media y la varianza. Puesto que el valor p entregado por las pruebas de permutaciones es exacto, no es posible obtener un intervalo de confianza.

En términos generales, las pruebas exactas de permutaciones para la diferencia entre dos grupos A y B (puede extenderse esta idea para más grupos) de tamaños n_A y n_B , respectivamente, sigue los siguientes pasos:

1. Calcular la diferencia entre el estadístico de interés observado para ambos grupos.
2. Juntar ambas muestras en una muestra combinada.
3. Obtener todas las permutaciones de la muestra combinada en que se pueden distribuir las observaciones en dos grupos de tamaños n_A y n_B .
4. Construir la distribución de las posibles diferencias, calculando la diferencia entre el estadístico de interés obtenido para ambos grupos en cada una de las permutaciones.
5. Calcular el valor p exacto, dado por la proporción de permutaciones en que el valor (absoluto, si es bilateral) de la diferencia calculada es menor/mayor o igual al valor (absoluto si es bilateral) de la diferencia observada.

Puesto que las pruebas exactas de permutaciones requieren calcular todas las permutaciones, solo resultan adecuadas para muestras pequeñas, pues requieren de una enorme cantidad de cálculos. En consecuencia, si la muestra es grande, suele tomarse una muestra aleatoria de las permutaciones posibles, procedimiento que suele denominarse **simulación de Monte Carlo**, y a partir de ella calcular un valor p aproximado dado por la ecuación 12.4.

Podemos ver que en la ecuación 12.4 se suma 1 tanto al numerador como al denominador. Esto corresponde a una corrección que debemos aplicar puesto que el método de Monte Carlo no es insesgado.

De los párrafos anteriores se desprende que las pruebas de permutaciones (exactas o no) son adecuadas para el contraste de hipótesis con dos o más muestras, pues determinan una significación estadística (valor p).

En términos generales, el procedimiento para efectuar una prueba de permutaciones usando simulaciones de Monte Carlo no es muy distinto al de bootstrapping, aunque hay algunas diferencias fundamentales en el trasfondo:

1. Formular las hipótesis a contrastar (e identificar el estadístico de interés θ).
2. Crear una gran cantidad P de permutaciones (generalmente terminada en 9 para simplificar los cálculos) a partir de las muestras originales, usando **muestreo sin reposición sobre la muestra combinada**, y obtener el estadístico θ para cada una de las muestras.
3. Generar la distribución que el estadístico θ tendría si la hipótesis nula fuese cierta.

4. Determinar la probabilidad de encontrar un valor de θ al menos tan extremo como el observado en la distribución generada.

Debemos fijarnos en que, a diferencia de bootstrapping, las pruebas de permutaciones usan muestreo sin reposición puesto que, si la hipótesis nula fuera cierta, cada permutación de los valores obtenidos en la muestra combinada sería igualmente probable. Así, lo que se hace en cada repetición es tomar una muestra sin repetición de la muestra original (es decir, “reordenar” las observaciones) y asignar aleatoriamente cada observación a uno de los grupos, respetando los tamaños n_A y n_B de las muestras originales.

12.2.1 Prueba de permutaciones para comparar una variable continua en dos muestras independientes

El profesor de una asignatura inicial de programación, que se imparte para estudiantes de primer año de Ingeniería y estudiantes de último año de otras carreras que pueden cursar dicha asignatura como electivo, desea estudiar si existen diferencias en el rendimiento académico de ambos grupos. Para ello, considera una muestra de $n_A = 20$ estudiantes de primer año de Ingeniería y $n_B = 12$ estudiantes de último año de otras carreras.

El profesor ha decidido comparar el promedio de calificaciones finales de ambos grupos, usando para ello una prueba de permutaciones con $P = 5999$ repeticiones y un nivel de significación $\alpha = 0,05$. La diferencia observada para las muestras originales es $\bar{x}_A - \bar{x}_B = -0,017$, sugiriendo que los estudiantes de Ingeniería tienen peores calificaciones. Así, las hipótesis a contrastar son:

Denotando como μ_A al promedio de calificaciones finales de estudiantes de primer año de Ingeniería en el curso inicial de programación bajo estudio, y como μ_B al promedio de calificaciones finales de estudiantes de último año de otras carreras en el mismo curso, entonces:

$$H_0: \mu_A - \mu_B = 0$$

$$H_A: \mu_A - \mu_B \neq 0$$

Tras hacer la prueba, la distribución generada se asemeja bastante a la normal, aunque con una ligera asimetría hacia la derecha (figura 12.10), y el valor p obtenido para el contraste de hipótesis es $p = 0,969$, por lo que concluye con 95 % de confianza que no existe diferencia entre las calificaciones finales de ambos grupos de estudiantes.

Intrigado por este resultado, pues el profesor tiene la fuerte sensación de que, en general, los estudiantes de Ingeniería tienen más calificaciones deficientes que los estudiantes de otras carreras, ha decidido hacer un nuevo estudio con las mismas muestras, comparando ahora la diferencia en la variabilidad (manteniendo la misma cantidad de repeticiones e igual nivel de significación). Así:

Denotando como σ_A a la varianza de las calificaciones finales de estudiantes de primer año de Ingeniería en el curso inicial de programación bajo estudio, y como σ_B a la varianza de las calificaciones finales de estudiantes de último año de otras carreras en el mismo curso, entonces:

$$H_0: \sigma_A - \sigma_B = 0$$

$$H_A: \sigma_A - \sigma_B \neq 0$$

La diferencia observada entre las varianzas de la muestra original es $\sigma x_A - \sigma x_B = 2,560$, sugiriendo que la variabilidad de las calificaciones obtenidas por los estudiantes de ingeniería es mayor. Tras efectuar el contraste de hipótesis, obtiene como resultado $p = 0,003$, evidencia suficiente para rechazar la hipótesis nula en favor de la hipótesis alternativa. Así, el profesor concluye que su percepción no es del todo errada, puesto que la variabilidad de las calificaciones es significativamente mayor para los estudiantes de Ingeniería.

Para hacer estos estudios, el profesor desarrolló en R el script 12.7. A pesar de que existen algunos paquetes de R para realizar pruebas de permutaciones, hemos decidido en esta ocasión implementar el procedimiento creando la función `contrastar_hipotesis_permutaciones()`, cuya especificación puede leerse en el script

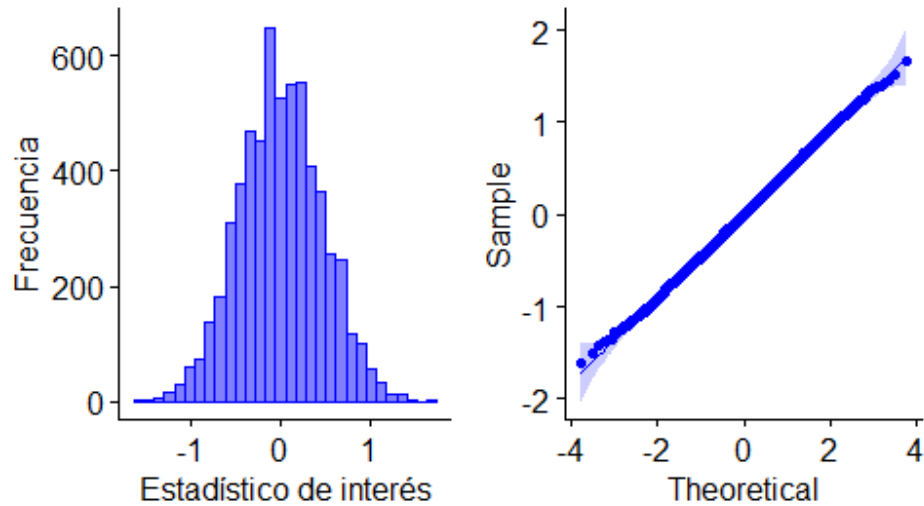


Figura 12.10: histograma y gráfico Q-Q de la distribución para la diferencia de medias generada mediante permutaciones.

12.7, la cual realiza el proceso y arroja como resultado el valor p resultante. Podemos ver que esta función opera usando como estadístico de interés la diferencia de un estadístico θ entre dos muestras y que la función que calcula dicho estadístico θ para una muestra se entrega como argumento.

Script 12.7: pruebas de permutaciones para variables numéricas.

```
1 library(ggpubr)
2
3 # Crear muestras iniciales.
4 a <- c(5.4, 4.7, 6.3, 2.9, 5.9, 5.1, 2.1, 6.2, 1.6, 6.7, 3.0, 3.3,
5       5.0, 4.1, 3.3, 3.4, 1.2, 3.8, 5.8, 4.2)
6
7 b <- c(4.0, 4.1, 4.3, 4.3, 4.3, 4.2, 4.3, 4.3, 4.4, 4.1, 4.3, 4.0)
8
9 # Establecer semilla y cantidad de repeticiones.
10 R = 5999
11 set.seed(432)
12
13 # Función para obtener una permutación.
14 # Argumentos:
15 # - i: iterador (para llamadas posteriores).
16 # - muestra_1, muestra_2: muestras.
17 # Valor:
18 # - lista con las muestras resultantes tras la permutación.
19 obtiene_permutacion <- function(i, muestra_1, muestra_2) {
20   n_1 <- length(muestra_1)
21   combinada <- c(muestra_1, muestra_2)
22   n <- length(combinada)
23   permutacion <- sample(combinada, n, replace = FALSE)
24   nueva_1 <- permutacion[1:n_1]
25   nueva_2 <- permutacion[(n_1+1):n]
26   return(list(nueva_1, nueva_2))
27 }
28
29 # Función para calcular la diferencia de un estadístico de interés entre las
30 # dos muestras.
31 # Argumentos:
```

```

32 # - muestras: lista con las muestras.
33 # - FUN: nombre de la función que calcula el estadístico de interés.
34 # Valor:
35 # - diferencia de un estadístico para dos muestras.
36 calcular_diferencia <- function(muestras, FUN) {
37   muestra_1 <- muestras[[1]]
38   muestra_2 <- muestras[[2]]
39   diferencia <- FUN(muestra_1) - FUN(muestra_2)
40   return(diferencia)
41 }
42
43 # Función para calcular el valor p.
44 # Argumentos:
45 # - distribucion: distribución nula del estadístico de interés.
46 # - valor_observado: valor del estadístico de interés para las muestras
47 #   originales.
48 # - repeticiones: cantidad de permutaciones a realizar.
49 # - alternative: tipo de hipótesis alternativa. "two.sided" para
50 #   hipótesis bilateral, "greater" o "less" para hipótesis unilaterales.
51 # Valor:
52 # - el valorp calculado.
53 calcular_valor_p <- function(distribucion, valor_observado,
54                             repeticiones, alternative) {
55   if(alternative == "two.sided") {
56     numerador <- sum(abs(distribucion) > abs(valor_observado)) + 1
57     denominador <- repeticiones + 1
58     valor_p <- numerador / denominador
59   }
60   else if(alternative == "greater") {
61     numerador <- sum(distribucion > valor_observado) + 1
62     denominador <- repeticiones + 1
63     valor_p <- numerador / denominador
64   }
65   else {
66     numerador <- sum(distribucion < valor_observado) + 1
67     denominador <- repeticiones + 1
68     valor_p <- numerador / denominador
69   }
70
71   return(valor_p)
72 }
73
74 # Función para graficar una distribución.
75 # Argumentos:
76 # - distribucion: distribución nula del estadístico de interés.
77 # - ...: otros argumentos a ser entregados a gghistogram y ggqqplot.
78 graficar_distribucion <- function(distribucion, ...) {
79   observaciones <- data.frame(distribucion)
80
81   histograma <- gghistogram(observaciones, x = "distribucion",
82                             xlab = "Estadístico de interés",
83                             ylab = "Frecuencia", bins = 30, ...)
84
85   qq <- ggqqplot(observaciones, x = "distribucion", ...)
86
87   # Crear una única figura con todos los gráficos de dispersión.
88   figura <- ggarrange(histograma, qq, ncol = 2, nrow = 1)
89   print(figura)
90 }

```

```

91
92 # Función para hacer la prueba de permutaciones.
93 # Argumentos:
94 # - muestra_1, muestra_2: vectores numéricos con las muestras a comparar.
95 # - repeticiones: cantidad de permutaciones a realizar.
96 # - FUN: función del estadístico E para el que se calcula la diferencia.
97 # - alternative: tipo de hipótesis alternativa. "two.sided" para
98 #   hipótesis bilateral, "greater" o "less" para hipótesis unilaterales.
99 # - plot: si es TRUE, construye el gráfico de la distribución generada.
100 # - ...: otros argumentos a ser entregados a graficar_distribucion.
101 contrastar_hipotesis_permutaciones <- function(muestra_1, muestra_2,
102                                               repeticiones, FUN,
103                                               alternative, plot, ...) {
104   cat("Prueba de permutaciones\n\n")
105   cat("Hipótesis alternativa:", alternative, "\n")
106   observado <- calcular_diferencia(list(muestra_1, muestra_2), FUN)
107   cat("Valor observado:", observado, "\n")
108
109   n_1 <- length(muestra_1)
110
111   # Generar permutaciones.
112   permutaciones <- lapply(1:repeticiones, obtiene_permutacion, muestra_1,
113                           muestra_2)
114
115   # Generar la distribución.
116   distribucion <- sapply(permutaciones, calcular_diferencia, FUN)
117
118   # Graficar la distribución.
119   if(plot) {
120     graficar_distribucion(distribucion, ...)
121   }
122
123   # Calcular el valor p.
124   valor_p <- calcular_valor_p(distribucion, observado, repeticiones,
125                               alternative)
126
127   cat("Valor p:", valor_p, "\n\n")
128 }
129
130
131
132 # Hacer pruebas de permutaciones para la media y la varianza.
133 contrastar_hipotesis_permutaciones(a, b, repeticiones = R, FUN = mean,
134                                   alternative = "two.sided", plot = TRUE,
135                                   color = "blue", fill = "blue")
136
137 contrastar_hipotesis_permutaciones(a, b, repeticiones = R, FUN = var,
138                                   alternative = "two.sided", plot = FALSE)

```

12.2.2 Prueba de permutaciones para comparar medias de más de dos muestras correlacionadas

Supongamos ahora que un estudiante de un curso de programación necesita comparar la eficiencia de tres algoritmos de ordenamiento: *quicksort*, *bubblesort* y *mergesort*. Para ello, ha seleccionado aleatoriamente

6 arreglos de igual tamaño y registrado para cada uno de ellos el tiempo de ejecución utilizado por cada algoritmo (en milisegundos) bajo iguales condiciones, como muestra la tabla 12.4.

Instancia	Quicksort	Bubblesort	Mergesort
1	11,2	15,7	12,0
2	22,6	29,3	25,7
3	23,4	30,7	25,7
4	23,3	30,8	23,7
5	21,8	29,8	25,5
6	40,1	50,3	44,7

Tabla 12.4: tiempos de ejecución para las diferentes instancias con cada algoritmo del ejemplo.

Tras comprobar mediante la figura 12.11 que no se cumple la condición de normalidad, el estudiante ha decidido usar permutaciones para resolver su problema. Para ello, ha considerado un nivel de significación $\alpha = 0,01$ y un total de 2999 repeticiones, obteniendo como resultado un valor $p = 0,0003$, mucho menor que el nivel de significación. En consecuencia, concluye con 99 % de confianza que el tiempo de ejecución promedio es significativamente diferente para al menos uno de los algoritmos.

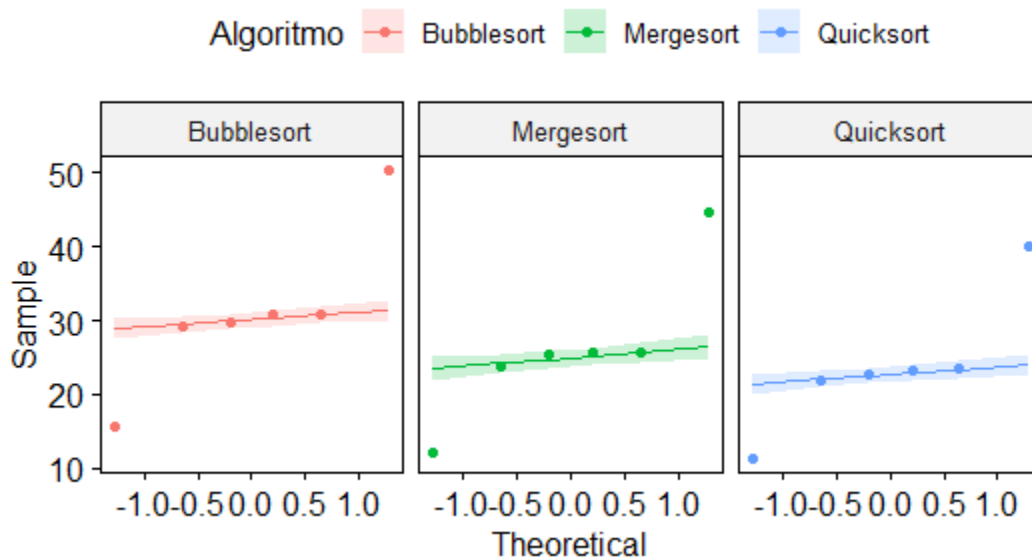


Figura 12.11: gráfico Q-Q para comprobar el supuesto de normalidad para el ejemplo.

A fin de determinar qué algoritmos difieren en su tiempo promedio de ejecución, ha decidido llevar a cabo un procedimiento post-hoc, calculando los valores p para las medias de las diferencias entre cada par de grupos para las diferentes permutaciones, obteniendo los resultados que se presentan en la figura 12.12. En consecuencia, el estudiante concluye con 99 % de confianza, que existen diferencias significativas en el tiempo promedio de ejecución entre los algoritmos Quicksort y Bubblesort y los algoritmos Bubblesort y Mergesort. Al estudiar las diferencias observadas, puede ver que Bubblesort es menos eficiente que los dos algoritmos restantes.

El script 12.8 corresponde a la solución desarrollada por el estudiante.

Script 12.8: prueba de permutaciones para muestras correlacionadas.

```

1 library(ggpubr)
2 library(ez)
3 library(tidyverse)
4

```

Análisis post-hoc (permutaciones) para la diferencia de las medias

Valores p:

Quicksort - Bubblesort: 0.000

Quicksort - Mergesort: 0.116

Bubblesort - Mergesort: 0.003

Diferencias observadas:

Quicksort - Bubblesort: -7.367

Quicksort - Mergesort: -2.483

Bubblesort - Mergesort: 4.883

Figura 12.12: resultado del procedimiento post-hoc.

```
5 # Crear el data frame.
6 Quicksort <- c(11.2, 22.6, 23.4, 23.3, 21.8, 40.1)
7 Bubblesort <- c(15.7, 29.3, 30.7, 30.8, 29.8, 50.3)
8 Mergesort <- c(12.0, 25.7, 25.7, 23.7, 25.5, 44.7)
9 Instancia <- factor(1:6)
10 datos anchos <- data.frame(Instancia, Quicksort, Bubblesort, Mergesort)
11
12 datos_largos <- datos anchos %>% pivot_longer(c("Quicksort", "Bubblesort",
13                                               "Mergesort"),
14                                               names_to = "Algoritmo",
15                                               values_to = "Tiempo")
16
17 datos_largos[["Algoritmo"]] <- factor(datos_largos[["Algoritmo"]])
18
19 # Verificar condición de normalidad.
20 g <- ggqqplot(datos_largos, "Tiempo", facet.by = "Algoritmo",
21               color = "Algoritmo")
22
23 print(g)
24
25 # Establecer nivel de significación.
26 alfa <- 0.01
27
28 # Obtener el valor observado, correspondiente al estadístico F entregado
29 # por ANOVA para la muestra original.
30 anova <- ezANOVA(datos_largos, dv = Tiempo, within = Algoritmo,
31                  wid = Instancia, return_aov = TRUE)
32
33 valor_observado <- anova[["ANOVA"]][["F"]]
34
35 # Generar permutaciones.
36 R = 2999
37 # copia_ancha <- data.frame(datos anchos)
38
39 set.seed(432)
40
41 # Función para obtener una permutación.
42 # Devuelve una matriz de datos con formato ancho.
43 obtiene_permutacion <- function(i, df_ancho) {
44   df_ancho[, 2:4] <- t(apply(df_ancho[, 2:4], 1, sample))
45   return(df_ancho)
46 }
```

```

47
48 # Obtiene permutaciones
49 permutaciones <- lapply(1:R, obtiene_permutacion, datos anchos)
50
51 # Función para obtener el estadístico F para una matriz de datos con formato
52 # ancho.
53 obtiene_F <- function(df_ancho) {
54   df_largo <- df_ancho %>% pivot_longer(c("Quicksort", "Bubblesort",
55                                           "Mergesort"),
56                                           names_to = "Algoritmo",
57                                           values_to = "Tiempo")
58
59   df_largo[["Algoritmo"]] <- factor(df_largo[["Algoritmo"]])
60
61   anova <- ezANOVA(df_largo, dv = Tiempo, within = Algoritmo, wid = Instancia,
62                   return_aov = TRUE)
63   return(anova[["ANOVA"]][["F"]])
64 }
65
66 # Genera distribución de estadísticos F con las permutaciones.
67 distribucion <- sapply(permutaciones, obtiene_F)
68
69 # Obtener valor p.
70 p <- (sum(distribucion > valor_observado) + 1) / (R + 1)
71 cat("ANOVA de una vía para muestras pareadas con permutaciones\n")
72 cat("p =", p, "\n\n")
73
74 # Análisis post-hoc.
75
76 # Función para calcular la media de las diferencias para dos columnas de una
77 # matriz de datos en formato ancho.
78 obtiene_media_difs <- function(df_ancho, columna_1, columna_2) {
79   media <- mean(df_ancho[[columna_1]] - df_ancho[[columna_2]])
80   return(media)
81 }
82
83 # Obtiene las las medias de las diferencias observadas
84 dif_obs_quick_bubble <- obtiene_media_difs(datos anchos, "Quicksort",
85                                           "Bubblesort")
86
87 dif_obs_quick_merge <- obtiene_media_difs(datos anchos, "Quicksort",
88                                           "Mergesort")
89
90 dif_obs_bubble_merge <- obtiene_media_difs(datos anchos, "Bubblesort",
91                                           "Mergesort")
92
93 # Obtiene las distribuciones de las medias de las diferencias permutadas
94 dist_medias_difs_quick_bubble <- sapply(permutaciones, obtiene_media_difs,
95                                         "Quicksort", "Bubblesort")
96
97 dist_medias_difs_quick_merge <- sapply(permutaciones, obtiene_media_difs,
98                                         "Quicksort", "Mergesort")
99
100 dist_medias_difs_bubble_merge <- sapply(permutaciones, obtiene_media_difs,
101                                         "Bubblesort", "Mergesort")
102
103 # Obtener valores p.
104 num <- sum(abs(dist_medias_difs_quick_bubble) > abs(dif_obs_quick_bubble)) + 1
105 den <- R + 1

```

```

106 p_quick_bubble <- num / den
107
108 num <- sum(abs(dist_medias_difs_quick_merge) > abs(dif_obs_quick_merge)) + 1
109 den <- R + 1
110 p_quick_merge <- num / den
111
112 num <- sum(abs(dist_medias_difs_bubble_merge) > abs(dif_obs_bubble_merge)) + 1
113 den <- R + 1
114 p_bubble_merge <- num / den
115
116 cat("\n\n")
117 cat("Análisis post-hoc (permutaciones) para la diferencia de las medias\n")
118 cat("-----\n")
119 cat("Valores p:\n")
120
121 cat(sprintf("Quicksort - Bubblesort: %.3f\n", p_quick_bubble))
122 cat(sprintf("Quicksort - Mergesort: %.3f\n", p_quick_merge))
123 cat(sprintf("Bubblesort - Mergesort: %.3f\n", p_bubble_merge))
124
125 cat("\nDiferencias observadas:\n")
126 cat(sprintf("Quicksort - Bubblesort: %.3f\n", dif_obs_quick_bubble))
127 cat(sprintf("Quicksort - Mergesort: %.3f\n", dif_obs_quick_merge))
128 cat(sprintf("Bubblesort - Mergesort: %.3f\n", dif_obs_bubble_merge))

```

12.3 EJERCICIOS PROPUESTOS

1. En tus palabras, ¿qué son las técnicas de remuestreo?
2. Explica si ¿podría considerarse que la prueba exacta de Fisher usa técnicas de remuestreo?
3. ¿En qué se parecen y en qué se diferencian las técnicas de bootstrapping y permutación?
4. En tus palabras, ¿qué es una simulación Monte Carlo?
5. ¿Cómo realizarías bootstrapping para determinar si la estatura media de estudiantes de Las Condes es igual a la estatura media de estudiantes de La Pintana?
6. ¿Cómo usarías Monte Carlo para verificar que un medicamento para bajar el colesterol funciona en un grupo de 30 personas escogidas al azar?
7. ¿Cómo usarías bootstrapping para analizar si tres algoritmos necesitan tiempos similares en procesar 25 instancias de prueba del problema de la mochila?
8. ¿Cómo usarías Monte Carlo para conocer si un medicamento para bajar la presión funciona al administrarlo a un grupo de personas hipertensas, comparando con un grupo de personas hipertensas recibiendo placebo y un grupo de control de personas no hipertensas?

REFERENCIAS

- Amat Rodrigo, J. (2016). *Resampling: test de permutación, simulación de Monte Carlo y Bootstrapping*. Consultado el 31 de mayo de 2021, desde https://www.cienciadedatos.net/documentos/23_resampling_test_permutacion_simulacion_de_monte_carlo_bootstrapping
- Hesterberg, T., Monaghan, S., Moore, D. S., Clipson, A. & Epstein, R. (2003). *Bootstrap Methods and Permutation Tests*. Consultado el 3 de junio de 2021, desde <https://statweb.stanford.edu/~tibs/stat315a/Supplements/bootstrap.pdf>